The Objective Sum Constraint.

Jean-Charles Régin, Thierry Petit

The Objective Sum Constraint

Jean-Charles Régin[1] and Thierry Petit[2]
jcregin@gmail.com; thierry.petit@mines-nantes.fr

[1]Université de Nice-Sophia Antipolis, I3S, CNRS.
[2]École des Mines de Nantes, LINA, CNRS.

Abstract. Constraint toolkits generally propose a sum constraint where a global objective variable should be equal to a sum of local objective variables, on which bound-consistency is achieved. To solve optimization problems this propagation is poor. Therefore, ad-hoc techniques are designed for pruning the global objective variable by taking account of the interactions between constraints defined on local objective variables. Each technique is specific to each (class of) practical problem. In this paper, we propose a new global constraint which deals with this issue in a generic way. We propose a sum constraint which exploits the propagation of a set of constraints defined on local objective variables.

1 Introduction

A lot of optimization problems aim to minimize (or maximize) the sum (or a scalar product) of some variables. The variable equals to that sum is usually called the *objective variable* and denoted by obj, and the sum is called the *objective constraint*. For convenience, we will call *sub-objective variables* the variables involved in the sum while the other variables but obj will be called *problem variables*. The sub-objective variables are denoted by $sobj_i$ and the problem variables by x_i.

Lower bounds of the objective variable are computed from the sum constraint by considering minimum values of the sub-objective variables. Thus, we usually state that $\sum \underline{sobj_i} = \underline{obj}$ where $\underline{y}$ denotes the minimum value in the domain of the variable y. The minimum value of the sub-objective variables are computed from the constraints (different from the objective constraint) which involve them.

This model has three main drawbacks:

- The filtering algorithm associated with the objective constraint is weak although it should deserve more attention because it is often the most important one.
- The fact that some problem variables may occur in several constraints, each involving a different sub-objective variable, is ignored.
- The fact that some sub-objective variables may be involved in several constraints (different from the objective constraint) is ignored while it is important [3, 4].

This paper tries to remedy to these drawbacks. The main idea is to change the way the sub-objective constraints are propagated. Consider, for instance, that we have the following problem to solve: $obj = sobj_1 + sobj_2$ has to be minimized while respecting the constraints $sobj_1 = 2x + y$ and $sobj_2 = z - x$ with x taking its value in $[0, 10]$, y in $[0, 10]$ and z in $[10, 20]$. The classical filtering algorithm will lead to $\underline{sobj_1} = 2*0+0 =$

0 and $\underline{sobj_2} = 10 - 10 = 0$, therefore $\underline{obj} = 0$. However, if we look at problem from the point of view of the variable x, this means that we need to find a value for x such that $sobj_1 + sobj_2$ is minimized with $sobj_1 = 2x + y$ and $sobj_2 = z - x$. Clearly, $x = 0$ minimizes the value of $sobj_1$ but not at all the value of $sobj_2$ and $x = 10$ minimizes the value of $sobj_2$ but not the value of $sobj_1$. If we try all values for x then we will discover that $x = 0$ minimizes the value of $sobj_1 + sobj_2$ which is equal to $y + z$ whereas any other value v will lead to $y + z + v$. Thus, by applying a stronger form of consistency we deduce that obj is greater than or equal to 10.

The main idea of this paper is to count for each problem variable involved in a constraint involving a sub-objective variable a lower bound of its contribution to the final objective. We can easily figure out how we can use the result obtained from one problem variable but it is unclear to see how the values obtained for all the problem variables can be sum up. In this paper, we propose a generic answer to this question, inspired from the PFC-MRDAC algorithm [2]: we select a problem variable x and compute a lower bound of the sum of the sub-objective variables involved in a constraint with x. Then we remove these sub-objective variables and we repeat the process for another variable. At the end we can sum up the different lower bounds of the subsums because there are disjoint, and we obtain a lower bound for the objective variable.

The paper is organized as follows. First, we propose a new lower bound of the objective variable. Then, we introduce a new filtering algorithm. At last, we give some related work and we conclude.

2 Objective Sum Constraint

Definition 1 *Given*

- obj *an objective variable*
- $SO = \{sobj_1, ..., sobj_p\}$ *a set of sub-objective variables,*
- $PX = \{x_1, ..., x_q\}$ *a set of problem variables disjoint from* SO*,*
- C_{SO} *be a set of constraints, each of them involving at least one sub-objective variable.*

The **objective-sum constraint** *is equivalent to the constraint network defined by the variables* $(obj \cup SO \cup PX)$ *and the constraints* C_{SO} *and the objective constraint* $obj = \sum_{sobj_i \in SO} sobj_i$

We will consider that we are provided with `minsobj`$((x, a), sobj_i, C)$, a function returning the minimum value of $sobj_i$ compatible with (x, a) on the constraint C. It returns the minimum value of $sobj_i$ consistent with C if x is not involved in C. This function can be implemented using the filtering algorithm of C and it does not have to be exact (a lower bound can be used).

3 Lower bounds

3.1 Multiple constraints involving a sub-objective variable

Consider a variable x involved in several constraints with the same sub-objective variable $sobj_i$ and a value a of $D(x)$. A lower bound of $sobj_i$ from the point of view of the

variable x is the maximum value of $\texttt{minsobj}((x,a), sobj_i, C)$ among all constraints involving $sobj_i$:

Property 1 *We define*

- $C_{SO}(sobj)$*, the set of constraints involving* $sobj$*.*
- $\underline{xsobj_i}(x,a) = \max_{C \in C_{SO}(sobj_i)}(\texttt{minsobj}((x,a), sobj_i, C))$
- $\underline{xsobj_i}(x) = \min_{a \in D(x)}(\underline{xsobj_i}(x,a))$

Then, $sobj_i \geq \underline{xsobj_i}(x)$

If we denote by $minseparate(sobj_i)$ the minimum value of $sobj_i$ consistent with each constraint of $C_{SO}(sobj_i)$ taken separately then we have:

Property 2 $\underline{xsobj_i}(x) \geq minseparate(sobj_i)$

3.2 Multiple sub-objective constraints involving a variable

The computation of the minimum value of the objective variable is usually made by independent computations of the minimum value of all the sub-objective variables. We propose here to take into account simultaneously some other constraints, that is, establishing a stronger form of consistency.

Consider a variable x involved in a constraint C_1 involving the sub-objective variable $sobj_i$ and in a constraint C_2 involving the sub-objective variable $sobj_j$. Considering the constraints separately means that the value of x consistent with the minimum value of $sobj_i$ on C_1 may be different than the one consistent with $sobj_j$ on C_2. So, for the sum $sobj_i + sobj_j$, we can compute a lower bound of the minimum value of this sum by considering different values of x, which is clearly an underestimation because in any solution the value of x will be the same. Instead, for each value a of x, we propose to compute a lower bound of $sobj_i + sobj_j$. There is a value (x,a) for which $l = sobj_i + sobj_j$ is minimum among all the values in the domain of x. Then, l is a new lower bound of $sobj_i + sobj_j$. Note that this value can never be less than the one computed for each sub-objective variable separately. Hence, we have:

Property 3 *Let* $S \subseteq SO$ *be a set of sub-objective variables. We define:*

- $L(S,(x,a)) = \sum_{sobj_i \in S} \underline{xsobj_i}(x,a)$
- $\underline{sumobj}(x,S) = \min_{a \in D(x)}(L(S(x,a)))$ *Then,*

$$\sum_{sobj_i \in S} sobj_i \geq \underline{sumobj}(x,S)$$

Property 4 *If* S *is equal to all the sub-objective variables involved in a constraint with* x *we have:*

$$\underline{sumobj}(x,S) \geq \sum_{sobj_i \in S} minseparate(sobj_i)$$

It is not relevant to consider sub-objective variables $sobj_i$ such that there is no constraint in $C_{SO}(sobj_i)$ involving x. Next section shows how SO can be partitioned so as to sum up efficiently several lower-bounds computed thanks to Property 3, in order to obtain a lower-bound of obj.

3.3 A new lower bound of the objective variable

Property 3 shows how we can improve the computation of some sums of sub-objective variables. We will obtain a new lower bound of the objective if the addition of that sub-sums is equal to the whole sum at the end. That is, if no sub-objective variable is added twice. By partionning the set of sub-objective variables SO we avoid counting twice the same variable. In addition we can apply for each part Property 3. Therefore, we have the following proposition:

Proposition 1 *Let SO be the set of sub-objective variables, and $\mathcal{P}(SO) = \{S_1, ..., S_k\}$ a partition of SO, and $\{x_1, ..., x_k\}$ a set of variables. Then,*

$$obj = \sum_{sobj_i \in SO} sobj_i \geq \sum_{S_i \in \mathcal{P}(SO)} \underline{sumobj}(x_i, S_i)$$

The main issue is to determine how the partition is defined and which variable is associated with each set. In fact, these questions are strongly related. We can imagine several techniques for computing this partition. For instance, here is a possible algorithm:

1. For each $x \in PX$ define $sobjvar(x)$, the set of sub-objective variables which are involved in a constraint involving x
2. Define O equal to SO and $lobj$ equals to 0
3. Select the variable x having the largest $sobjvar(x) \cap O$ cardinality.
4. Compute $\underline{sumobj}(x, sobjvar(x) \cap O)$ and add the result to $lobj$
5. Remove $\overline{sobjvar}(x)$ from O
6. Repeat from step 3) until there is no more variable in O

At the end of this algorithm $lobj$ is a new lower bound of obj.

4 Filtering algorithm

From the lower bound presented in Section 3 we can design a filtering algorithm whose goal is to reduce the domain of some problem variables and not only the domains of the objective or sub-objective variables. We will consider two types of problem variables: those that are associated with a part of $\mathcal{P}(SO)$ and those that are not.

Consider a variable x and S be the part of $\mathcal{P}(SO)$ associated with it. If x is assigned to $b \neq a$ then the lower bound is increased by $L(S, (x, b)) - \underline{sumobj}(x, S)$. If this increment is too much for the objective variable (i.e. the lower bound is greater than $\overline{obj}$) then the value (x, b) is not consistent with the constraint. Thus we have:

Property 5 *For any value b of x we define:*

- $slack((x, b), S) = L(S, (x, b)) - \underline{sumobj}(x, S)$
- $K = \overline{obj} - [\sum_{S_i \in \mathcal{P}(SO)} \underline{sumobj}(x_i, S_i)]$

Then, each value (x, b) such that $slack((x, b), S) > K$ **is not consistent** *with the objective constraint.*

Once the new lower bound has been computed this filtering algorithm does not cost anything more because all the sums $\sum_{sobj_i \in S} \underline{xsobj_i}(x, b)$ have been computed for all the values of x.

With respect to variables which are not associated with a part, we can either ignore them (or compute a new partition and apply the filtering algorithm).

5 Application

The objective sum constraint arises frequently in problem with multi-objectives. For instance, it occurs in multidimensional bin packing problems (in which an item has several dimensions), like the one appearing in the cloud computing management. In these problems, we need to fill in servers (bins) with virtual machine (items) while respecting all the sums of the capacities, one for each dimension. In the same time, we associate with each dimension and with each server a cost representing the assignment of the virtual machine to the server. For instance, a dimension may be the disk access time and the cost represents a penalty depending of the time access given by the server. The objective is to minimize the sum of all the penalties and penalties may be involved in several side constraints.

6 Related Work

The idea of the partition comes from PFC-MRDAC, an algorithm for solving over-constrained problems. The original version of the algorithm dealing only with binary constraints is given in [2]. A simpler presentation, not restricted to binary constraints, is given in [5]. Our generic technique is an alternative to some dedicated algorithms for propagating sum constraints, which are specific to particular classes of optimization problems (e.g., see [1] for constraint-based scheduling).

7 Conclusion

This paper presented a preliminary work about a new filtering algorithm for the objective sum constraint, which is useful in the resolution of multi-objective problems.

References

1. A. Kovács and J. C. Beck. A global constraint for total weighted completion time for cumulative resources. *Constraints, in print.*, 2010.
2. J. Larrosa, P. Meseguer, T. Schiex, and G. Verfaillie. Reversible DAC and other improvements for solving Max-CSP. *Proceedings AAAI*, pages 347–352, 1998.
3. T. Larsson and M. Patriksson. On side constrained models of traffic equilibria. In *Proc. of the 19th Course of the International School of Mathematics*, pages 169–178. Plenum Press, 1995.
4. T. Petit and E. Poder. Global propagation of side constraints for solving over-constrained problems. *Annals of Operations Research*, 184:295–315, 2011.
5. J.-C. Régin, T. Petit, C. Bessière, and J.-F. Puget. An original constraint based approach for solving over constrained problems. In *Proc. of CP'00*, pages 543–548, Singapore, 2000.